Monday update
Bernhard Egger authored 5 months ago

cdbe0309

H02.Software.Installation.md 14.93 KiB

Hands-on: Setting up the Development Environment

In this hands-on, we set up the work environment we are going to use throughout this course.

You will

- install and setup the Gretchen VM and
- install the necessary software for interacting with Gretchen
- install and configure an editor for your work

Installing the Gretchen VM

The Robot Operating System (ROS) we will be using in this course requires a Ubuntu 20.04 LTS (Focal Fossa) system. If you do not run Ubuntu 20.04, we recommend to install the Gretchen VM on your system. Note that non-Intel systems (such as M1/M2 Apple notebooks) are unable to run the VM.

1. Install VirtualBox 6.1.44 on your machine

   - Gentoo Linux:

     ```
     sudo emerge -avu '=app-emulation/virtualbox-6.1.44' '=app-emulation/virtualbox-modules-6.1.44' '=app-er
     ```

   - Direct links: Windows, OS X, 64-bit Linux, Ubuntu 22.04, Ubuntu 20.04, Debian 11, openSUSE 15.3/4.
   - For other systems, visit the VirtualBox 6.1 Download Page

2. Install the VirtualBox 6.1.44 Extension Pack
   After downloading the extension pack, start VirtualBox and select File → Preferences → Extensions and add the extension pack to VirtualBox.
   If you prefer working in a terminal, you can alternatively run

3. Download the Gretchen VM from Google Drive
   Gretchen VM (6.2 GiB)
   Checksums:
   md5: c5ba1df4ee7e14a6ce95ae22c6ad85ee
   sha256: 4d0cb8dac6a1edb7355ac15556ef128b6ac6964cea1d484affa23b11c0073806

4. VirtualBox → Import, select downloaded Gretchen VM image
   Adjust hardware settings (such as the number of processor cores and the available memory) to match your laptop

5. Settings → Shared Folders → Add
   Folder Path: select a folder that you want to use to share files between the host and the VM
   Folder Name: host **(must be an exact match)**

6. If you have a HiDPI (4K) screen: set scaling factor
   Settings → Display → Scale Factor: 200%

7. Start VM and login
   The user `developer` is logged in automatically. The default password is `gretchen`, feel free to change it with the command `passwd` in a shell.

8. Set keyboard layout and screen resolution
   xfce4 Menu → Settings → Display
   xfce4 Menu → Settings → Keyboard → Layout

9. Shutdown

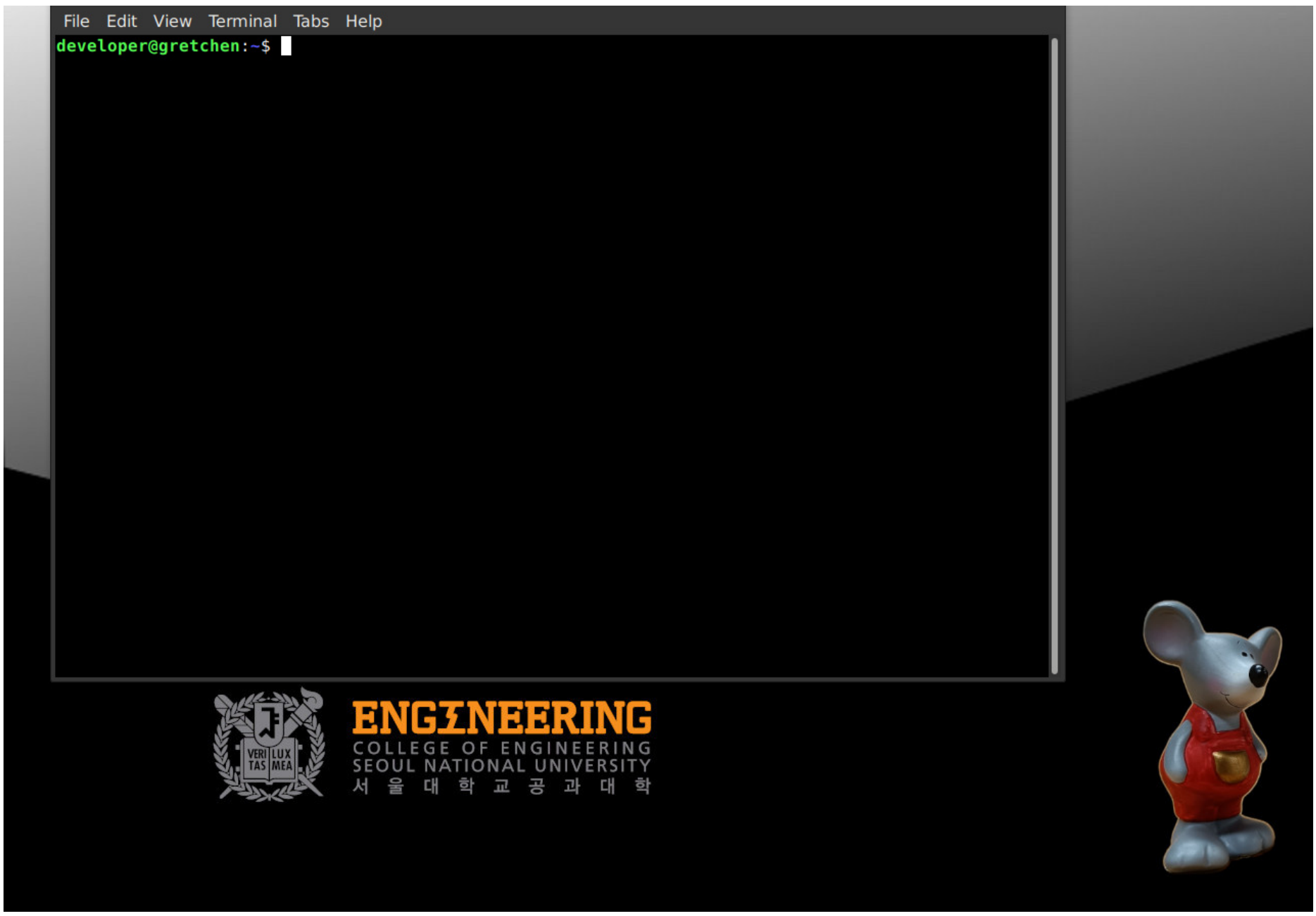
10. Create a snapshot after configuring your VM

You can delete the downloaded Gretchen VM image to free up some space.

System Configuration

The Terminal

We will work a lot in a terminal, so it is important that you get familiar with it. You can open a new terminal by clicking the terminal icon in the lower panel (second icon from left) or by going through the menu (top-left of screen) Applications → Terminal Emulator.



Git Setup

We use the GitLab server on our CSAP server to hand-out lecture materials. We need to our global identity (provide the same email you used to sign up at CSAP GitLab).

In a terminal, execute the following commands:

```
git config --global user.name "Your Name"
git config --global user.email your@email.com
```

You can cache your password for a given amount of seconds so you won't have to enter it every single time

```
git config --global credential.helper 'cache --timeout=3600'
```

Alternatively, you can use the `store` credential helper - this will store you password in clear-text and once and for all

```
git config --global credential.helper 'store'
```

Obviously, this would be a very bad idea on a public/shared machine.

Installation of ROS - the Robot Operating systems

The Robot Operating System (ROS) is an open-source collection of software libraries and tools to build robot applications. Gretchen and its software stack is built on top of ROS. In the following, we install

Install ROS

Open a terminal and execute the follwing command

```
wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_noetic.sh
```

Let's make that command executable

Now, we can execute the script. The script downloads ROS directly from the ROS servers and installs it in our VM.

```
./install_ros_noetic.sh
```

Follow the instructions of the installation script. The script requires elevated privileges and will ask you for your password. If you haven't changed it yet, the password is `gretchen`.

Depending on your network speed and the performance of your VM / computer, this may take quite some time. After the installation has finised, make sure to close your terminal and open a new one before continuing.

```
exit
```

Install System and ROS libraries

Now, let's install a few more system and ROS libraries that are required to control and interact with Gretchen. In a new terminal, execute the following command

```
sudo apt install -y python3-pip libv4l-dev ros-noetic-ddynamic-reconfigure ros-noetic-realsense2-description
```

The command must run with elevated privileges and will thus ask you for your password. Then answer 'Y'ES to the question whether you want to install the listed packages.

Again, close your terminal now and open a new one before continuing.

OpenCV, IMUtils, and Dlib

In our experiments, we will use Python libraries for computer vision (OpenCV), image manipulation (IMutils), and deep learning (Dlib). OpenCV was already pulled in during the installation of ROS, and so we only need to install IMUtils and Dlib.

Install IMUtils via pip

The IMUtils library is installed with pip, Pythons package installer:

```
sudo pip install imutils
```

Install Dlib

We compile and install Dlib directly from the GitHub repository.

First, we download Dlib onto our machine. Make sure to execute all commands in sequence.

```
mkdir ~/lib
cd ~/lib
git clone https://github.com/davisking/dlib.git
cd dlib
```

Next, we configure and build Dlib

```
mkdir build
cd build
cmake .. -DDLIB_USE_CUDA=0 -DUSE_AVX_INSTRUCTIONS=1
cmake -build .
cd ..
```

Finally, we install Dlib through the provided install script

```
sudo python3 setup.py install --no DLIB_USE_CUDA
```

This may take some time on slower machines.

First Test

Let's test whether all libraries were installed successfully. Open a terminal and start Python

You will see Python print some information, then present you with its prompt `>>>`.

```
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

At the prompt ('>>>'), enter the following commands

```
import cv2
import imutils
import dlib
print("CV2: {}, imutils: {}, Dlib: {}".format(cv2.__version__, imutils.__version__, dlib.__version__))
```

You should get the following output

```
CV2: 4.2.0, imutils: 0.5.4, Dlib: 19.24.99
```

Great, all libraries have been installed successfully. Now exit Python with

```
quit()
```

Installing the Gretchen Development Environment

Now we need to install Gretchen's development environment. This environment contains the necessary drivers and bindings to ROS and also our example code and some exercises.

Cloning the the Repository

Gretchen's development environment is hosted on our CSAP GitLab server. We need to place the contents in the directory `~/catkin/src`. Execute the following commands

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone https://teaching.csap.snu.ac.kr/first-steps-in-programming-a-humanoid-ai-robot/isp-2023/gretchen.gi
```

You will need to provide your user credentials (email and password) for the Gitlab server at teaching.csap.snu.ac.kr. Note that you must apply for and be granted access to the ISP 2023 repository to be able to download the development environment.

Installing the Development Environment

Now that the repository has been cloned, we need to compile and install it. Execute the follwing commands

```
cd ~/catkin_ws/src/gretchen/DynamixelSDK/python
sudo python3 setup.py install
```

Followed by

```
cd ~/catkin_ws
catkin_make
```

The last command will take some time to complete.

Configuring the Development Environment

Last, we copy a file with udev rules which ensures that Gretchen's camera always appears under the same name.

```
sudo cp ~/catkin_ws/src/gretchen/config/55-cam-motor.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

There are many good code editors from very simple ones to large integrated development environments (IDEs) that offer hundres of features and take up gigabytes of space.

We recommend two editors for this course: Vim and PyCharm. Vim is a representative of the former group of editor - lean, fast, executed directly in the shell, but extremely powerful and extensible with plugins. Vim is installed by default on most Linux systems and used by many "hard-core" Unix programmers. PyCharm, on the other hand, is a full-fledged IDE and probably a bit easier to use for beginners.

The Vi IMproved Text Editor

Real programmers use vim. The main idea behind vi and Vi IMproved is that programmers can perform all editing tasks - including commands such as moving through source code, copy-paste, search/replace, and so on - with the keyboard. In other words, by using vim, you will rarely have to move your hand from the keyboard to the mouse and back.

Vim by itself is an extremely powerful editor, however, it lacks many of the fancy features of modern IDEs. In the following, you will install a couple of plugins that turn vim (almost) into a full-fledged IDE.

vim-fugitive plugin

The Fugitive plugin display the status of files in a Git repository in Vim's status line and can also be used to execute Git commands.

Installation

```
mkdir -p ~/.vim/pack/tpope/start
cd ~/.vim/pack/tpope/start
git clone https://github.com/tpope/vim-fugitive.git/
vim -u NONE -c "helptags fugitive/doc" -c q
```

Edit ~/.vimrc to contain the following

```
"-------------------------------------------------------------------------------
" Fugitive plugin: show Git status in statusline
"
hi CVSStatus cterm=bold ctermbg=LightGray ctermfg=DarkBlue
hi StatusLine cterm=NONE ctermbg=LightGray ctermfg=Black
set laststatus=2
set statusline=
set statusline+=%#CVSStatus#%{FugitiveStatusline()}
set statusline+=%#StatusLine#\ %<%f\ %h%m%r%=%-14.(%l,%c%V%)\ %P
```

tagbar plugin

The tagbar plugin displays a handy overview of the edited file on the right-hand side of the window for easy navigation.

First, we install ctags, a library that indexes language objects found in source code files

```
sudo apt install -y ctags
```

The plugin is installed by

```
mkdir -p ~/.vim/pack/majutsushi/start
cd ~/.vim/pack/majutsushi/start
git clone https://github.com/majutsushi/tagbar.git
vim -u NONE -c "helptags tagbar/doc" -c q
```

Edit ~/.vimrc to contain the following

```
"-------------------------------------------------------------------------------
" tagbar plugin configuration
"
let g:tagbar_width=24
autocmd VimEnter * nested :call tagbar#autoopen(1)
autocmd FileType * nested :call tagbar#autoopen(0)
autocmd BufEnter * nested :call tagbar#autoopen(0)
```

Modify `g:tagbar_width` as you please

The YouCompleteMe plugin enables source code completion and inline API help for vi.

Installation

```
mkdir -p ~/.vim/bundle
cd ~/.vim/bundle
git clone https://github.com/ycm-core/YouCompleteMe.git
cd YouCompleteMe
git submodule update --init --recursive
python3 install.py --clangd-completer
```

The YCM plugin contains several submodules and may take some time to compile.

Edit ~/.vimrc to contain the following

```
"------------------------------------------------------------------------------
" youcompleteme plugin configuration
"
set rtp+=~/.vim/bundle/YouCompleteMe
autocmd BufRead,BufNewFile * setlocal signcolumn=yes
```

Help with Vim

Editing in vi is quite difficult for new users because most are not familiar with the command/editing mode separation and vi's many keyboard shortcuts. Once you have mastered it, however, you will never want to go back to the pointy-pointy-clicky-clicky interfaces used by modern IDEs. If you still prefer to point and click, maybe you are in luck: vi is so popuplar that many IDEs offer a "vi editing mode" that emulates vi's command and editing functionality. Familiarize yourself with vi by going through one of the many vi tutorials. We recommend the tutorial of the University of Washington Using vi, the Unix Visual Editor.

Since you won't be able to memorize all the different vi commands at once, a cheat sheet is most helpful. Among the many good cheat sheets, vim.rtorr.com is quite useful.

PyCharm

PyCharm is an IDE designed specifically for Python programming, providing advanced features and tools to streamline code editing, debugging, and project management. You can install PyCharm via Ubuntu's Snap Store (Applications → System → Snap Store) or via the command line:

```
sudo snap install pycharm-community --classic
```

After the installation has completed, you will find PyCharm in Applications → Development → PyCharm Community Edition.

Finishing Off the Installation

Congratulations, you have now a working environment to program and interact with Gretchen! Before we continue, we recommend taking a snapshot of your Gretchen VM so that you can go back to this state in case something goes wrong. Shut down your VM, then select Snapshot → Take in the VirtualBox GUI.